Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 52 (2015) 1010 – 1015

Procedia Computer Science

International Workshop on Big Data and Data Mining Challenges on IoT and Pervasive Systems (BigD2M 2015)

An Outlier Detect Algorithm using Big Data Processing and Internet of Things Architecture

Alberto M. C. Souza[a], José R. A. Amazonas[b]

[a]*Escola Politécnica, University of São Paulo - USP and BANDTEC College, St. Estela - 268, São Paulo 04011-001, Brazil*
[b]*Escola Politécnica, University of São Paulo - USP, Av. Prof. Luciano Gualberto, 380, São Paulo, 05508-010, Brazil*

Abstract

Data management in the Internet of Things is a crucial aspect. Considering a world of interconnected objects which constantly exchange many kinds of information, the volume of generated data and involved processes, implies that data management becomes critical. The aim of this paper is to propose an outlier detection procedure using the K-means algorithm and Big Data processing using the Hadoop platform and Mahout implementation integrated with our chosen Internet of Things architecture.

Peer-review under responsibility of the Conference Program Chairs
Keywords: Internet of Things, Big Data, Architecture, Outlier

1. Introduction

The Internet of Things (IoT) is a new communication paradigm in which the Internet is extended from the virtual world to interface and interact with objects of the physical world. A huge amount of applications and services can then be developed and simultaneously an immense set of challenges must be overcome to make the IoT come true. IoT involves different areas of knowledge as pervasive computing, network communication, object identification and, in special, data processing. In this context we introduce pattern recognition mechanisms in the IoT architecture [1].

The focus of the paper is the implementation of an algorithm to detect outliers using Big Data processing, to integrate it in the chosen and implemented IoT architecture. The modular architecture implementation enables an easy introduction of other algorithms according to the needs of new applications and services.

The paper is organised as follows: after this brief Introduction, in Section 2 we describe the main IoT concepts, the K-means algorithm and Big Data technology. The proposed architecture, its implementation details and experimental validation are shown in Section 3. Conclusions and future works are presented in Section 4.

* Corresponding author. Tel.: +55-11-5574-6844; fax: +55-11-5574-6844.
E-mail address: alberto.souza@bandtec.com.br

Peer-review under responsibility of the Conference Program Chairs
doi:10.1016/j.procs.2015.05.095

2. Background

In this section we review the main concepts related to the paper: Internet of Things, K-means algorithm and Big Data.

2.1. The Internet of Things concept

As stated in[2], Internet of Things is a global network infrastructure, linking physical and virtual objects through the exploitation of automatic identification, data capture and communication capabilities. This infrastructure includes the existing and evolving Internet and other network developments. It will offer specific object-identification, sensor and connection capability as the basis for the development of independent federated services and applications. These will be characterized by a high degree of autonomous data capture, event transfer, network connectivity and interoperability, actuation and control.

According to the CASAGRAS inclusive model, a real-world object has its identification ID and associated information stored on some kind of item-attendant data carrier as, for example, on a RFID tag. It is important to realize that the identification technology is not restricted to RFID. Biometry and bar codes are other examples of ID technology that can be employed. The information is retrieved from the object by means of an interrogator that acts as a gateway device and sends it to be stored in a host management system. The Internet is used both to allow access to the retrieved information and to search for further information and associated applications and services. The end result is that an action will take place either displaying new information and/or acting upon the object and/or the environment[3]. The whole process is context-aware and the final action depends on the object itself and its present status in the current environment.

2.2. The K-means algorithm

The K-means algorithm, proposed by MacQueen in[4], is a clustering algorithm based on a similarity measure between objects. It works as follow: the algorithm receives a parameter indicating the number k of clusters and represented by their centroids $s_i, 1 \leq i \leq k$; it also receives N random objects or observations. In each iteration, each object is allocated to a cluster determined by the shortest distance between the object and all centroids. After each iteration the algorithm relocates the centroids by minimizing the mean distance of all objects in the cluster to its centroid. When the centroids positions have stabilized the algorithm has converged.

2.3. Big Data processing with Hadoop and Mahout

Sun and Heller in[5] mention that Big Data refers to large datasets that are difficult to store, search, view, represent and analyze. Smith in[6] states that Big Data refers to extremely large datasets that cannot be processed and/or analyzed by conventional tools. Big Data requires large computational power to efficiently process such large datasets within reasonable times. This technology involves massive parallel processing databases (MPP), data mining grids, distributed file systems, cloud computing, the Internet and scalable store systems.

Hadoop is the term used to refer to a family of related projects that fall under the umbrella of infrastructure for distributed computing and large-scale data processing. According to White in[7], Hadoop is best known for its implementation of MapReduce and its distributed file system implementation.

The MapReduce is a distributed data processing model and execution environment that runs on large clusters of commodity machines. The MapReduce breaks the processing into map and reduce phases and each phase is based on key/value pairs used as input and output. The programmer also specifies two functions, the map and the reduce functions, to be used in the specific implementation[7].

Hadoop comes with a Hadoop Distributed File System called HDFS that is a file system designed to store very large files with streaming data access patterns, running on clusters of commodity or common hardware platforms[7]. The MapReduce and HDFS have an application programming interface to enable developments and use of their functionalities.

Other relevant project to this paper is the Mahout, that is an open source machine learning library from Apache. According to Owen et al. in[8], Mahout aims to be the machine learning tool of choice when the collection of data to

be processed is very large, perhaps far too large for a single machine. Mahout's implementation is written in Java and built upon Apache's Hadoop distributed computation project.

Mahout has several algorithms implementations of classification and clustering, being its K-means implementation the one of interest to our work.

3. Outlier detection algorithm with Big Data processing and Internet of Things architecture

According to Tan et al. in[9], proximity-based approaches can be used in anomaly detection. Angiulli and Basta in[10], Lei et al. in[11], Jiang et al. in[12] used clustering algorithms in outlier detection. In our implementation we adopt this approach to propose the outlier detection algorithm associated with Big Data processing.

Our implementation has 5 steps:

1. The application inputs the raw data to create a clustering model.
2. Run the Canopy Clustering algorithm[13] on the initial data to propose an initial value of the number K of centroids, using the Mahout's implementation proposed in[8].
3. Run the K-means algorithm, starting with the centroids proposed by the Canopy algorithm, to create a model of clusters also using the Mahout's implementation proposed in[8].
4. Get the information about the clusters and their centroids and radii generated by the K-means execution;
5. With these values the method *isOutLier* can be used. This implementation calculates the Euclidean[14] distance of the instance parameter to all centroids, and if is greater then each radius, this instance is classified as an outlier.

Figure 1 illustrates the proposed approach.

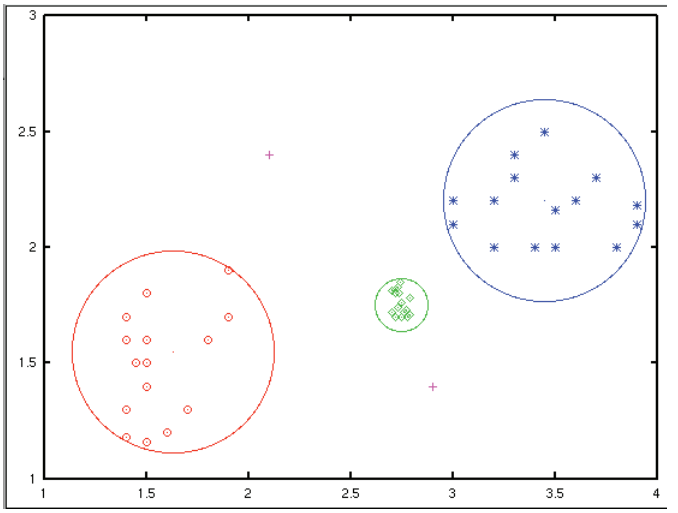

Fig. 1. Outlier detection: three clusters along their radii are shown and two outlier points were detected.

In Figure 1 three clusters have been generated: cluster (a) represented by red circles, cluster (b) represented by green diamonds and cluster (c) represented by blue stars, and two outliers points represented by magenta plus, as they are outside the clusters' circles defined by the clusters' radii.

3.1. The Outlier algorithm and Internet of Things middleware

To integrate the implemented algorithm, we extend the *LinkSmart* Internet of Things middleware, developed in the Hydra Project[15] by introducing a new module in the middleware. Figure 2 shows the proposed architecture implemented according to the layer structure of the LinkSmart middleware and the class diagram that represents the developed classes.

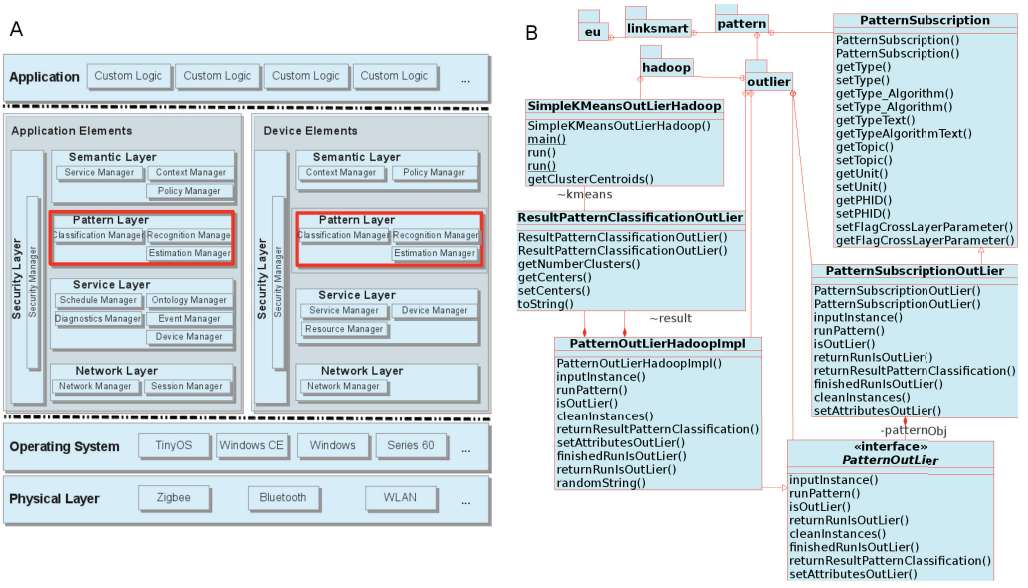

Fig. 2. (A) The LinkSmart middleware's [15] new layer structure incorporating pattern recognition mechanisms. (B) The class diagram that represents the implemented classes and the developed oriented programming structure.

In Figure 2(A) we see a box designated by *Pattern Layer*, highlighted by a red rectangle. This new layer has three managers: classification, recognition and estimation, which implement the pattern recognition functionalities. At the current stage of this research the implementation focused on the application elements seen at left side of Figure 2(A) and the classification manager hosts the outlier detect algorithm.

In Figure 2(B) we see the class diagram that represents the implemented classes and the oriented programming structure developed in the project. The classes *PatternSubscription* and *PatternSubscriptionOutLier* allow the integration with the LinkSmart middleware and implement a new service to receive instances and process data. The class *PatternOutLier* defines the methods to a class to be a pattern outlier class integrated with the LinkSmart. The main class is the *PatternOutLierHadoopImpl* that implements the integration with Hadoop and delegates the clustering algorithm implementation to the *SimpleKMeansOutLierHadoop*, which by its turn returns all clusters and their radii to the main class, along the created model. Finally, any instance can be submited to the algorithm to calculate if it is an outlier or not. It is important to realise that the oriented programming implementation allows future new implementations of this functionality also to be integrated with the LinkSmart.

The algorithms to estimate values, classify and recognize behaviors, and to detect outliers [14 16] contribute to network traffic reduction in the IoT context as the upper application layers will not receive raw data anymore but pre-processed information by the LinkSmart middleware pattern services. The focus of this paper is the implementation of an outlier detect algorithm with Big Data processing in the middleware layer, validate this implementation and the proposed integration with an IoT architecture.

3.2. A testbed implementation

The raw data used in the resource manager are from the Guildford's facility proposed in the European Smart Santander Project [17].

The retrieved data were inserted in the Mysql [18] database and a class to simulate the resource manager was created. The resource manager provides temperature and light intensity values from a single sensor node, designated as *node25*. At this stage the experiment is a proof of concept and processes data from a single sensor, but the proposed architecture is prepared to process and manage large databases according to the big data concept.

The client application implemented has two functions: (i) it is a client of the pattern manager; and (ii) uses the pattern manager as a coordinator to control execution of the outlier detect algorithm. Figure 3 shows the execution

of the client application and the Receiver Operating Characteristic (ROC) curve using the results generated by the execution of the outlier detection algorithm on the provided raw data.

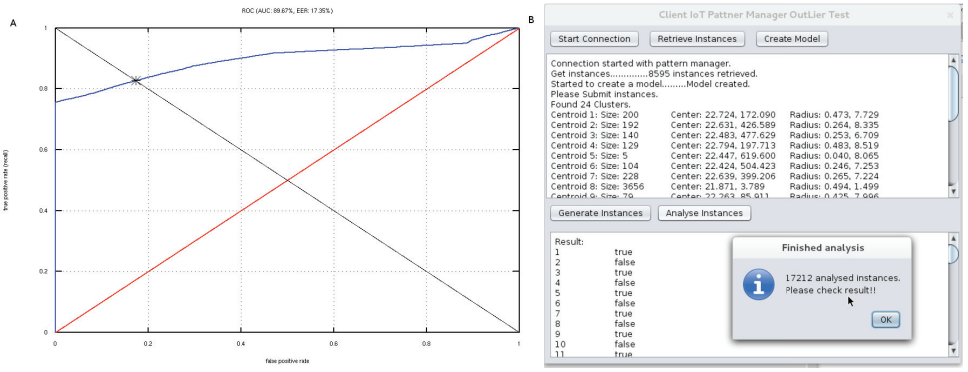

Fig. 3. (A) A result of Outlier detection algorithm execution on the real data and (B) Execution of the application client test developed.

In Figure 3(A) we see the ROC curve of the outlier detect algorithm execution. This ROC provides the Area Under Curve (AUC) obtained by the classification algorithm execution: a good classifier must have AUC greater than 0.5. In this case the obtained value was a 0.8967 area.

Figure 3(B) shows the execution of the client test application presenting the coordinator function and the result of the classifier. This application starts the connection with the LinkSmart middleware, retrieves and inserts all instances in the pattern manager and, finally, creates a model with real instances. Next, the application generates the new instances to be analysed, submits each instance to the created model and the results are displayed in the interface.

In this test the model was created from 8595 real instances, 8595 outlier instances have been artificially created by taking each real instance and generating a new instance by adding a random number to the real one. The set with real and artificial instances has been submitted to the analysis by the outlier detection algorithm. The result shows all real and outlier instances with their respective classification (*true* or *false*). This information was used to generate the ROC curve.

This execution refers to temperature and light intensity data values obtained on 2014-02-01 between 00:00:00 and 23:59:59.

We have also tested the algorithm using data values obtained along all days of February 2014. The data set of each day has been analysed and the results had minimum, maximum and mean AUC values respectively equal to 0.8456, 0.9465 and 0.8762. It can be seen that the algorithm has always performed extremely well.

An application has been developed to fulfil two purposes: (i) to implement a coordinator in charge of triggering the creation of a model by the pattern manager; (ii) to attest the integration of the pattern manager with the LinkSmart middleware architecture. The use of the outlier detection algorithm makes the application receive all instances without outliers and eliminates the overhead to analyse the raw data. This implementation sends filtered information to any application that needs them contributing to reduce the processing at the application level, to increase the energy efficiency of the whole system and to alleviate scalability issues when the number N of client applications increase.

4. Conclusions and future work

In this work we have implemented an outlier detect algorithm using the Big Data technology, namely the Hadoop Framework and Mahout K-means algorithm implementation. This algorithm runs integrated with the IoT architecture implemented by the LinkSmart middleware. Its scalability is ensured by the use of Big Data technology enabling physical objects and sensors to be directly plugged in the middleware. In fact, it provides great scalability allowing the creation of clusters with hundreds or thousands of Hadoop instances that can be plugged transparently into the LinkSmart and client applications. The object-oriented structured programming allows other implementations to be plugged in the extended LinkSmart middleware.

The proposed architecture and its implementation enhance the functionality and potential of use of the IoT LinkSmart middleware. This framework addresses scalability, contextualisation and flexibility enabling a huge number of different kinds of devices to acquire environment context awareness. The information provided by a single light sensor, for example, can be read by various applications without any interference on each other. The raw data is processed only once in the middleware layer, so different applications may be simpler and receive the filtered information, without the need to process the original raw data. This approach reduces the network traffic and the overall energy consumption.

The testbed implementation validated the proposed algorithm and the integration with IoT architecture using real data from the Smart Santander Project. The execution shows that the IoT architecture implementation, the LinkSmart middleware and the pattern recognition algorithms implemented in the middleware layer work with real data. The ROC curve shows the good quality of the results produced by the proposed outlier detection algorithm.

As future work we intend to validate and evaluate the proposed architecture with large databases according to the big data concept. We will implement an estimation algorithm integrated with the IoT architecture. In addition we intend to propose and validate cross layer communication in the IoT architecture.

Acknowledgements

We acknowledge the ICT- 2009-257992 (SmartSantander) and the REDUCE project grant EP/I000232/1 under the Digital Economy Programme run by Research Councils UK that supported the development and deployment of the SmartCampus testbed.

References

1. Souza, A.M., Amazonas, J.R.. A novel smart home application using an internet of things middleware. In: *Smart Objects, Systems and Technologies (SmartSysTech), Proceedings of 2013 European Conference on*. 2013, p. 1–7.
2. CASAGRAS, E.F.P.. Casagras final report: Rfid and the inclusive model for the internet of things. *EU FP7 Project CASAGRAS* 2009;.
3. Amazonas, J.R.d.A.. Network virtualization and cloud computing: Iot enabling thecnologies. *Casagras2 Academic Seminar* 2011;URL: `http://www.casagras2.com.br/downloads/day2/2-Jose_Roberto_de_Almeida_Amazonas-Network_Virtualization_and_Cloud_Computing_IoT_enabling_echnologies.pdf`.
4. MacQueen, J.. Some methods for classification and analysis of multivariate observations. In: *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*. 1967, p. 281–297.
5. Sun, H., Heller, P.. Oracle information architecture: An architect s guide to big data. In: *An Oracle White Paper in Enterprise Architecture*. 2012, .
6. Smith, I.. *The Internet of Things 2012: New Horizons*. CASAGRAS2; 2012. ISBN 9780955370793. URL: `http://www.internet-of-things-research.eu/pdf/IERC_Cluster_Book_2012_WEB.pdf`.
7. White, T.. *Hadoop: The Definitive Guide*. O'Reilly Media, Inc.; 1st ed.; 2009. ISBN 0596521979, 9780596521974.
8. Owen, S., Anil, R., Dunning, T., Friedman, E.. *Mahout in Action*. Greenwich, CT, USA: Manning Publications Co.; 2011. ISBN 1935182684, 9781935182689.
9. Tan, P.N., Steinbach, M., Kumar, V.. *Introduction to Data Mining, (First Edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.; 2005. ISBN 0321321367.
10. Angiulli, F., Basta, S., Pizzuti, C.. Distance-based detection and prediction of outliers. *Knowledge and Data Engineering, IEEE Transactions on* 2006;**18**(2):145–160.
11. Lei, D., Zhu, Q., Chen, J., Lin, H., Yang, P.. Automatic k-means clustering algorithm for outlier detection. In: Zhu, R., Ma, Y., editors. *Information Engineering and Applications*; vol. 154 of *Lecture Notes in Electrical Engineering*. Springer London. ISBN 978-1-4471-2385-9; 2012, p. 363–372.
12. Jiang, M., Tseng, S., Su, C.. Two-phase clustering process for outliers detection. *Pattern Recognition Letters* 2001;**22**(67):691 – 700. URL: `http://www.sciencedirect.com/science/article/pii/S0167865500001318`.
13. McCallum, A., Nigam, K., Ungar, L.H.. Efficient clustering of high-dimensional data sets with application to reference matching. In: *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*; KDD '00. New York, NY, USA: ACM. ISBN 1-58113-233-6; 2000, p. 169–178.
14. Duda, R.O., Hart, P.E., Stork, D.G.. *Pattern Classification (2nd Edition)*. Wiley-Interscience; 2 ed.; 2000. ISBN 0471056693.
15. Sarnovsky, M., Kostelink, P., Butka, P., Hreno, J., Lackova, D.. First demonstrator of hydra middleware architecture for building automation. *Hydra Project* 2005;URL: `http://www.hydramiddleware.eu/`.
16. Theodoridis, S., Koutroumbas, K.. *Pattern Recognition, Fourth Edition*. Academic Press; 4th ed.; 2008. ISBN 1597492728, 9781597492720.
17. Nati, M., Gluhak, A., Abangar, H., Headley, W.. Smartcampus: A user-centric testbed for internet of things experimentation. In: *Wireless Personal Multimedia Communications (WPMC), 2013 16th International Symposium on*. 2013, p. 1–6.
18. Tahaghoghi, S., Williams, H.. *Learning MySQL*. O'Reilly Media; 2006. ISBN 9781449303969.